(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 166 703 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
08.07.2015 Bulletin 2015/28

(51) Int Cl.:
H04L 12/24 *(2006.01)* *H04L 29/12* *(2006.01)*

(21) Application number: **09170876.8**

(22) Date of filing: **21.09.2009**

(54) **Method and system for managing a hierarchical information base with an application layer protocol**

Verfahren und System zur Verwaltung einer hierarchischen Informationsbasis mit einem Anwendungsschichtprotokoll

Procédé et système pour la gestion d'une base d'informations hiérarchiques dotée d'un protocole de couches d'application

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO SE SI SK SM TR

(30) Priority: **22.09.2008 IN CH23062008**

(43) Date of publication of application:
24.03.2010 Bulletin 2010/12

(73) Proprietor: **Hewlett-Packard Development Company, L.P.**
Houston, TX 77070 (US)

(72) Inventors:
- **Sarkar, Sujoy**
  743127, West Bengal (IN)
- **Hyerle, Robert**
  38053, GRENOBLE (FR)

(74) Representative: **Lawman, Matthew John Mitchell**
EIP
Fairfax House
15 Fulwood Place
London, WC1V 6HU (GB)

(56) References cited:
US-A1- 2003 208 609 **US-A1- 2005 080 886**

- **MAZUMDAR S: "DIRECTORY ENABLED MANAGEMENT INFORMATION BASE - INTEGRATION OF MIB WITH DIRECTORIES USING COSNAMING AND JNDI" IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS:OPERATIONS AND MANAGEMENT, XX, XX, 25 August 2000 (2000-08-25), pages 1-32, XP002145963**

EP 2 166 703 B1

Description

Background

[0001] The Simple Network Management Protocol (SNMP) is a known application layer protocol that facilitates the exchange of management information between network devices.

[0002] SNMP is used in network management systems to monitor network-attached devices. It consists of a set of standards for network management, including an Application Layer protocol, a database schema, and a set of data objects.

[0003] Due to it's simplicity, SNMP has limitations with respect to information models and managed entity model descriptions. For example, SNMP has no notion of object behaviour, association or hierarchy in managed object descriptions.

[0004] US 20030208609 A1 discloses a method for configuring a network device which includes converting data from a first protocol to a second protocol, for example, from SNMP into LDAP, and vice versa.

[0005] US 20050080886 A1 discloses a proxy agent adapted to mediate between a network management system (NMS) that understands a first management protocol such as SNMP, and a network element that understands a second management protocol, such as Transaction Language 1 (TL1). The proxy agent comprises a directory for storing data components to be converted between the two management protocols. Each data component stored in this directory is associated both with a first data component identifier compatible with the first management protocol, and with a second data component identifier compatible with the second management protocol. The directory m ay be an LDAP directory in which the distinguished name of each directory entry may be used to provide one of the data component identifiers.

[0006] The paper: "DIRECTORY ENABLED MANAGEMENT INFORMATION BASE - INTEGRATION OF MIB WITH DIRECTORIES USING COSNAMING AND JNDI" by MAZUMDAR S (IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS:OPERATIONS AND MANAGEMENT, XX, XX, 25 August 2000 (2000-08-25), pages 1-32, XP002145963) describes using the CORBA naming service to restructure a SNMP MIB as a sub-tree of an enterprise wide directory. This is done in a way which enables information to be retrieved using string based naming whereby directory APIs such as LDAP API can retrieve information.

Brief Description of the Drawing

[0007] For a better understanding of the invention, embodiments will now be described, purely by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a schematic illustration of an SNMP-managed network employing an embodiment.

Detailed Description of the Invention

Simple Network Management Protocol

[0008] Referring now to Figure 1, an SNMP-managed network 10 consists of three key components: managed devices 12, agents 14, and one or more network-management systems (NMSs) 16.

[0009] A managed device 12 is a network node that contains an SNMP agent 14 and that resides on a managed network 10. Managed devices 12 collect and store management information and make this information available to a NMS 16 using SNMP. Managed devices 12, may sometimes be referred to as network elements, and can be routers and access servers, switches and bridges, hubs, computer hosts, or printers for example.

[0010] An agent 14 is a network-management software module that resides in a managed device 12. An agent 14 has local knowledge of management information and translates that information into a form compatible with SNMP.

[0011] An NMS 16 executes applications that monitor and control managed devices 12. NMSs typically provide the majority of processing and memory resources required for network management. One or more NMSs must exist on any managed network.

[0012] In essence, SNMP agents 14 expose management data on the managed devices 12 as variables. The managed devices are monitored and controlled using four basic SNMP commands: get, set, trap and traversal operations.

[0013] The get command is used by an NMS 16 to monitor managed devices 12. Using the get command, and NMS 16 examines different variables that are maintained by the managed devices 12.

[0014] The set command is used by an NMS 16 to control managed devices 12 since it enables the NMS 16 to change the values of variables stored within managed device 12.

[0015] The trap command is used by managed devices 12 to (asynchronously) report events to the NMS 16. For example, when certain types of events occur, a managed device 12 sends trap to the NMS.

[0016] Traversal operations are used by the NMS 16 to determine which variables a managed device 12 supports and to sequentially gather information in variable tables.

[0017] The variables accessible via SNMP are organized in hierarchies described by Management Information Bases (MIBs)

[0018] Each of the managed devices 12 in Figure 1 comprises a MIB 18 which is a collection of information that is organized hierarchically in a (virtual) database. A MIB 18 is accessed using a network-management protocol such as SNMP and generally comprises managed objects identified by object identifiers (OIDs).

[0019] A managed object (sometimes called a MIB object, or an object) is one of any number of specific characteristics of a managed device 12. Managed objects typically consist of one or more object instances, which are essentially variables.

[0020] Two types of managed objects exist: scalar and tabular. Scalar objects define a single object instance, whereas tabular objects define multiple related object instances that are grouped in MIB tables. An example of a managed object is "atInput", which is a scalar object that contains a single object instance, the integer value that indicates the total number of input AppleTalk packets on a router interface.

[0021] An object identifier (or object ID) uniquely identifies a managed object in the MIB hierarchy. Structurally, an object ID consists of a node in a hierarchically-assigned namespace, formally defined using the ITU-T's ASN.1 standard.

[0022] The MIB hierarchy can be depicted as a tree with a nameless root, the levels of which are assigned by different organizations. Top-level MIB object IDs belong to different standards organizations, while lower-level object IDs are allocated by associated organizations. Vendors can define private branches that include managed objects for their own products. MIBs that have not been standardized typically are positioned in the experimental branch. Successive numbers of the nodes, starting at the root of the tree, identify each node in the tree. The root of the tree contains the following three "arcs": 0 = ITU-T; 1 = ISO; 2 = joint-iso-itu-t. Thus, by way of example, the managed object "atInput" can be uniquely identified either by the object name - iso.identified-organization.dod.internet.private.enterprise.cisco.temporary_variables.AppleTalk. atInput-or by the equivalent object descriptor, 1.3.6.1.4.1.9.3.3.1.

Lightweight Directory Access Protocol

[0023] The Lightweight Directory Access Protocol (LDAP) is an application layer protocol for querying and modifying directory services running over TCP/IP.

[0024] Conventional LDAP deployments tend to use Domain Name System (DNS) names for structuring the topmost levels of the hierarchy. Deeper inside the directory might appear entries representing people, organizational units, printers, documents, groups of people or anything else that represents a given tree entry (or multiple entries).

[0025] LDAP is defined in terms of ASN.1, and protocol messages are encoded in the binary format Basic Encoding Rules (BER). It uses textual representations for a number of ASN.1 fields/types, however.

[0026] The LDAP protocol accesses LDAP directories containing directory "entries". An entry consists of a set of attributes. An attribute has a name (an attribute type or attribute description) and one or more values. The attributes are defined in a schema. For example, when represented in LDAP Data Interchange Format (LDIF) (since LDAP is a binary protocol as mentioned above), an entry can look like:

```
dn: cn=John Doe,dc=example,dc=com
cn: John Doe
givenName: John
sn: Doe
telephoneNumber: +1 888 555 6789
telephoneNumber: +1 888 555 1232
mail: john@example.com
manager: cn=Barbara Doe,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top.
```

[0027] Each entry has a unique identifier referred to as its Distinguished Name (DN). A DN consists of its Relative Distinguished Name (RDN) constructed from some attribute(s) in the entry, followed by the parent entry's DN. Thus, a DN may be thought of as analogous to a full path to a file and the RDN as a relative filename in a folder.

[0028] So, for the example provided above, dn is the name of the entry. It is not an attribute nor part of the entry. "cn=John Doe" is the entry's RDN, and "dc=example,dc=com" is the DN of the parent entry, where dc denotes Domain Component. The other lines show the attributes in the entry.

[0029] According to an embodiment there is provided a method of using SNMP to manage a hierarchical information base comprising a plurality of entries each defining object attributes. The method comprises the step of mapping each of a plurality of objects of an SNMP MIB to a unique attribute of the information base, thereby establishing, for each of the plurality of objects, a correspondence with a unique attribute, and vice-versa.

[0030] Thus, SNMP MIBs may be defined for the SNMP "get", "set" and "trap" commands with respect to a LDAP directory service, for example.

[0031] In particular, embodiments enable mapping between SNMP MIBs and hierarchical information bases. By way of example, embodiments enable Distinguished Names (DNs) of an LDAP directory service to be mapped to and/or from object IDs in SNMP MIBs.

[0032] Furthermore, embodiments may allow the use of SNMP to view Information Models, such as OpenCall Information Models (OCIMs), which are similar to database schemas that describe the structure and content of data. Embodiments may also allow the use of SNMP to view instances of information models, an instance being a particular set of objects and attributes (for example, database contents).

[0033] It will therefore be appreciated that embodiments address the limitations of SNMP thereby enabling SNMP to be used for the management of hierarchical information bases. Embodiments may therefore be implemented in the SNMP-managed network of Figure 1, for example in the NMS 16 and/or one or more of the managed devices 12.

[0034] In a preferred embodiment a function is adapted to undertake the translation between SNMP and LDAP object attributes based on the mapping definitions, and such a function resides in an NMS 16. The mapping definitions can be stored in the Information Model as part of a model instance or can be stored in some other form (XML, object-oriented language such as Java). If the mapping definition is not in same form as the information model, it may be preferable not to store it in the information model, but instead locate the mapping definition elsewhere in the managed network.

[0035] The mapping between DN's and OID's is based on what are hereinafter referred to as prototype DN's.

[0036] As explained above, DN's are composed of an RDN followed by the parent entry's DN, or alternatively, of a "left-hand-side" (lhs) and "right-hand-side" (rhs) pair. For example, a DN of "Instance=one;Application=myApp" has two types: "Instance" and "Application". In an embodiment, these types are used to build a structure, hereinafter referred to as a prototype DN defining a part of the OID to which the DN will be mapped.

[0037] A prototype DN is a DN. It is referred to as a "prototype" because it defines a set of potential DN's. Prototype DN's are used as part of the mapping definition. They describe that each lhs (at the given position in the DN) in the DN's within a set will have an associated SNMP sub-objectid.

[0038] For example: a prototype DN of "Instance=2;Application=3" defines a set of possible DN instances in a given model instance (database) will have the form: "Instance=<value>;Application=<value>" and that an SNMP sub-sequence of OID values will be "... 3.2 ..." This can then be combined with a prefix OID segment that is typically fixed for a given situation (i.e. the standard SNMP prefix, the vendor ID, etc , 1.3.....3.2). This objectID segment identifies an object type from which scalar or columnar data could be further indicated.

[0039] At each level in the prototype DN (e.g. at "Application=3") there is an associated object with attributes. It is this object that completes a mapping giving the index mapping method, the default value (described below).

[0040] When two prototype DN's share the same path from the root of the information model, they need to be disambiguated. Continuing the above example, suppose there is a prototype DN "Engine=10;Instance=2;Application=4" yielding an OID segment of "4.2.10". After traversing "4.2.x", it would be unclear if this refers to a scalar value in "Instance" or an object type for "Engine".

[0041] To avoid looking forward in OID strings and, in particular when indexing is introduced, it is required that the OID be unambiguous at every step of the traversal of sub-identifiers. The proposed solution for this example is to ensure that the OID sub-identifiers for all of "Instance's" descendants be distinct.

[0042] Descendants of a particular RDN are defined as child RDN types (lhs) and the attributes (scalars/columars) within the object attached to the RDN. B

[0043] To further explain, a pair of unambiguous prototype DN's is as follows:

```
Instance=1;Application=4
Engine=10;Instance=1;Application=4
```

[0044] In proposed embodiments, only a single OID may map to a given instance DN and only a single OID sub-sequence may map to a given prototype DN.

[0045] These prototype DNs described above are stored as part of the information model stored in a management server such as an information model management service (IMMS). It should also be appreciated, however, that the prototype DNs could be stored in another format in other locations (for example, in the filesystem, embedded in the mapping/gateway code, or at the web server, etc.).

[0046] Hence, whilst, in embodiments, they are referred to as "prototype DNs", they are viewed externally as DN's referring to object instances. The instances are used to define the mapping between SNMP scalar objects and information model object attributes within a given class. Such mapping from an SNMP object to information model attributes may be defined by a class definition "SNMPMap" as follows:

className - (string) - the name of the Information model class definition for objects attached at a DN "type". This may be redundant with some information Model descriptions.
isIndexed - (Boolean) - true if the RDN is indexed.
singletonValue - (string) - expected RDN value (rhs) if the definition is a singleton (i.e. not indexed)
attributeNames [MULTI_VALUE] - (string) - list of attributes within className that are defined within the MIB
attributeIds [MULTI_VALUE] - (integer) - list of OID sub-identifiers corresponding to the attributeNames
defaultAttribute [OPTIONAL] - (string) - attribute to use when translating a DN to an OID and no explicit attribute is provided
keys [MULTI_VALUE] - (string) - list of attributes within className used to compose the SNMP index
mapper [OPTIONAL] - (string) - name of a method that maps RDN's/instances of this node to OID index subsequences

[0047] Therefore, continuing with the above example again, the "Engine" object type might be defined as:

```
classname: "EnginePerf"
attributeNames: "speed" "temp" "state"
attributeIds: 1, 2, 3
```

[0048] Disregarding indexing which is described below, the OID for engine temperature would be "...4.1.10.2.0" Note the trailing zero is SNMP convention to indicate a scalar value.

[0049] From this description, it will be understood that

an MIB's definition files can be generated by hand or automatically. Also, given an information model description, a MIB object type mapping description can be constructed by hand or automatically.

[0050] The prefix OID segment can be defined as an attribute in the SNMPEnv class as follows:

```
OIDPrefix - (string) - the prefix of all
OID's within the implementation.
```

[0051] In summary, an OID is comprised of four sequences of sub-identifiers:

Prefix - fixed for a given implementation and comprised of sub-ids normally starting from iso (1), org (3), dod (6), internet (1), private (4), enterprises (1), ...

Type - a sequence of sub-ids derived from the Left-Hand Side (lhs) of the DNs. There is a one to one correspondence between a given lhs and a sub-id.

Attribute - a single sub-id that is either a singleton or a column in a table. If the attribute is a column, then the above type sequence has an additional sub-id or "1" appended to it. This corresponds to the SNMP convention of defining a table "Entry" for each table type. This implies that no "type" sub-id should have a value of "1" if its parent type has an associated table.

Index - a sequence of sub-ids derived from the index computations ("mapper") performed at each RDN along a given DN. The index has zero sub-ids when the RDN is not indexed.

Mapping using RDN Right-Hand Sides (RHSs)

[0052] Given that the mapping to object type has been detailed, it now remains to define how the correspondence between OID's and DN's/attributes is made for a particular instance of an information model. From the example above, it is possible that a model instance would include the following object instance DNs:

```
Instance=one;Application=myApp
Instance=one;Application=theirApp
Instance=twelve;Application=theirApp
Engine=v8;Instance=one;Application=myApp
Engine=v6;Instance=one;Application=myApp
```

[0053] Here, there are three (leaf) object classes: one each for Instance, Engine, and Application. There are two instances of Application (myApp and theirApp); three instances of Instance (myApp/one, theirApp/one, theirApp/twelve); and two instances for Engine (v8, v6). The "values" (or "right-hand-sides" rhs) of the RDN's can be used as indexes to the correct SNMP object instance. Notationally, the reference to the temperature of the v8 engine is "... 4.1.10.2.(myApp).(one).(v8)".

[0054] There are methods of "expanding" the values of non-integers such as strings and IP addresses into a sequence of sub-identifiers. The default method of mapping an RDN rhs is to treat it as a variable string and encode it as a length followed by the character codes. If the mapper attribute is not defined, this default is used. However, this is often undesirable.

[0055] Thus, to address the problematic use of the default mapping of indexes into OID sub-identifiers, the mapper attribute in the SNMPMap class definition specifies an alternate mapping function to use. One such mapper function will treat RDN rhs as integer values to be used as the index.

[0056] It may be the case that a given RDN within a prototype DN will always have exactly one instance: a singleton. In this case, indexing is not needed and only the correspondence between the RDN lhs and an OID sub-identifier is used. This is indicated by isIndexed being false and the expected value of a singleton's rhs is given by the singletonValue attribute. There should never be more than one rhs for such RDN's

Mapping using attribute values

[0057] A mapper function can be based on a sequence of attribute values within the object instance itself (a key) rather than on the RDN right-hand-side. Just as RDN's distinguish individual object instances, so must the keys. Further, such keys should typically remain stable over the life of the object instance: i.e. write once.

[0058] The sequence of attributes used for a key can be specified in a SNMPMap class definition as follows:

keys [MULTI_VALUE] - (string) - list of attributes within className used to compose the SNMP index.

[0059] Using such a definition, a specified mapper function can map attribute values to an OID sub-sequence representing the indexes.

DN/OID translation

[0060] A DN plus attribute name presented to an SNMP gateway (or some other application) at runtime for translation to an OID has either pre-loaded the SNMP prototype model or uses the model on demand. For each type (lhs) in the DN, the gateway is adapted to find the corresponding OID sub-identifier. The gateway also stores in memory the value (rhs) of each RDN element for use in forming the index. The mapper, if present, is then obtained. This process continues until the end of the DN.

[0061] If an attribute name is not provided, a pre-defined default attribute name- may be used.

[0062] After constructing this part of the OID, the named attribute's subidentifier is retrieved from attributelds.

[0063] For each RDN that is indexed, the OID subidentifier sequence is obtained using the appropriate mapper function. For example:

- a default string mapper using the rhs of the RDN is

employed when the mapper attribute is set to "default" or not defined;

- an integer mapper using the rhs of the RDN is employed when the mapper attribute is set to "intRhs"; and
- an attribute mapper using a sequence of attribute values is employed when the mapper attribute set to "attrs".

[0064] It is noted that when using the RDN rhs, the information model does not have to be accessed because the computation can be completed using only the DN. However, when the mapping is based on attribute values, the information model is accessed and the attribute values retrieved (for each RDN with this type of mapping).

[0065] For the reverse mapping, from OID to DN plus attribute, the SNMP gateway is arranged to: 1. Parse and remove global prefixes; 2. Using a mapping derived from the SNMP prototype description, recover the DN type names from the OID sub-identifiers (this process continues until a scalar/columar is found); 3. At each point during the traversal of the OID, memorise the mapper if it is present (i.e. isIndexed); 4. Use the OID subidentifier corresponding to a scalar/columnar to recover the attribute name using attributeIds and attributeNames; and 5. Using either the default or explicitly defined mappers, parse the indexes and convert to DN values. These values are used to "fill out" the DN.

[0066] When the mapping is based on attribute values, the SNMP gateway is adapted to lookup the object instance having the sequence of attribute values corresponding to the OID subidentifier sequence. Only then can it obtain the RND rhs. The mapper function performs this match and/or provides the OID sub-sequence to attribute value sequence.

[0067] Again, when the mapping is based on attribute values, the information model is accessed and the object instance name retrieved (for each RDN with this type of mapping).

OID dipping and default attribute mapping - "sets"

[0068] DN's and OID's are complete or "full" names. Often the more "global" or significant part of the name may be known by the context and/or a relative name is desired. In SNMP, the "type" and global prefix may be known and only the index values are needed to identify an instance. In the mapping of DN's to OID's for "on the fly" translation for "get", "set", and "trap" operations, one can indicate that the value of the OID is, or should be, "clipped." That is, only the sequence of index values is provided by the management system (for "set"). Similarly, the value returned to the management system (for "get" and "trap") will be only the sequence of index values.

Defining trap MIB's

[0069] The definition of SNMP trap MIB's is currently not stored in the IMMS, but in the SNMP gateway. At present, only some mapping information is represented in the IMMS.

[0070] Trap MIB's contain only information that can be derived from the SAF notification. MIB's themselves must be defined as well as the mapping between SAF notifications and the MIB. The mapping between SAF Notification Class Identifiers (NCI's) and SNMP trap MIB names is stored in the IMMS and is described below.

[0071] The "notification object" and "notifying object" present in the SAF notification are DN's. These are converted to OID's as per the above mapping definitions such that an SNMP manager can interrogate them. However, a DN alone does not specify a valid OID: an attribute must also be specified (OID's must, after parsing and indexing result in a reference to a simple type). The SAF notification does not contain an attribute name. Hence, the definition of the MIB provided to the gateway or in the information model defined mapping must specify the attribute that is used to obtain a valid OID. This attribute is translated to the sub-identifier of the scalar or columnar within the OID (which occurs prior to the index portion of the OID). This must be done for all DN's that are mapped to OID's for inclusion in the trap. However, this is not required if the OID is clipped as only the index portion of the OID is returned.

[0072] It is highly desirable that these attributes indicate the alarm state or other state (for alarm and state change notifications).

[0073] The NCI, if present, in a notification is used to access the IMMS mapping. If it is not present or not defined, default trap MIB's for alarm and state change are used. The NCI is composed of three integer values:

- vendor id
- major id
- minor id

[0074] The information model server contains a submodel that selects a particular notification definition:

"MinorID=k;MajorID=j;VendorID=i"
where "i, j, k" are integer values. The objects in the DN hierarchy at the "VendorID" and "MajorID" level do not contain any information except the key values used in the look up: that is, the values "i" and "j" respectively. These keys can be found using the DN names or, for convenience, as attribute values: named "id".

[0075] For example, the following attributes can be defined in a NotificationMap class definition:

MIBOID - (string) - the OID of the SNMP MIB definition to use when translating this notification. The OID

is relative to a global prefix that is defined within SNMPEnv ("OIDPrefix")
trimNotification -(Boolean) - true if only the index portion of the notification object OID should be included in the trap
trimNotifying - (Boolean) - true if only the index portion of the notifying object OID should be included in the trap
notificationAttr - (string) - used along with the mapped notification object DN to form a valid OID. Only necessary if no triming is done.
notifyingAttr - (string) - used along with the mapped notification object DN to form a valid OID. Only necessary if no triming is done.
id - (integer) - the MinorID value of this NotificationMap instance

Mapping SAF notifications to SNMP traps (req. 14)

[0076] The selection of trap MIB is based firstly on SAF "Event Type" which is a mandatory parameter. However, this is very generic and should only be used as a "fallback" mapping. In general, the "Notification Class Identifier" should be used to select the correct trap MIB. All X.73x mandatory parameters should be included in the trap and mappings need to be defined (for alarms and state changes) for:

- event type
- notification object DN
- notifying object DN (default is notification object)
- event time
- event ID
- probable cause
- perceived severity
- correlated event ID's
- changed state attribute list
- changed state attribute id, value

Example

[0077] The following examples are written in the Information Model Management Service (IMMS) native model description.

1. Below is an example of a hierarchically structured (LDAP) information base where there are six classes. From the root level, instances of these types appear as:

- OcSacAppT

-- OcSacSubscriberGroupT
-- OcSacCallStatisticsT
-- OcSacTelecomOperatorT

--- OcSacTelecomNetworkT

- OcSacCountryT

[0078] Thus, it will be appreciated that instances of OcSacCountryT have no descendants in the LDAP tree, OcSacAppT instances have three types of direct descendants, and OcSacTelecomOperatorT instances have one type of direct descendant.

2. According to an embodiment, mapping the above LDAP information base to an SNMP representation, where all six of the above classes along with their LDAP structure are mapped, is as follows:

OcSacApp- Has a type sub-id of "3" and is indexed on the appId attribute.
OcScaSubscriberGroup - A type sub-id of "2" and is indexed on subscriberGroupId attribute.
OcSacCallStatistics - A type sub-id of "3" and is indexed on subscriberGroupId attribute
OcSacTelecomOperator - A type sub-id of "4" and is indexed on operatorId attribute.
OcSacTelecomNetwork - A type sub-id of "3" and is indexed on an integer RDN right-hand side.
OcSacCountryT - A type sub-id of "2" and is indexed using a variable length string (default mapping)

[0079] The global environment defines an OID prefix value of "1.3.6.1.4.1.11.2", and a single trap is defined as follows:
[0080] Overload - Has a trap mid OID of "1.3.6.1.4.1.11.2.100.4.13" (prefix + MIB OID) and is mapped to an NCI of (44,10,13) (vendorId,majorId,minorId).

3. The classes defined in the information model for providing SNMP mapping are provided in a single file: snmp_classes.sf

4. The complete model of the six OcSac classes along with some sample instances are provided in a single file: OCSAC_all.sf

5. The complete LDAP information base of the SNMP mappings for the example is defined in a single file: SNMP_all.sf. Thus, this single file contains the class definitions for an SNMP to IMMS mapping (snmp_classes.sf), and the model structure describing the hierarchy of the information (OCSAC_all.sf). Such a file can be loaded by the IMMS and become part of its contents (they are transformed into an internal format and so may not be directly stored as files in the IMMS). The IMMS, in turn, resides in one or more SNMP agents. Alternatively, the file can be represented in Java, XML, or any suitable object-oriented language, and stored any stored at any suitable place(s) in a managed network.

6. Finally, the joint model: example definitions, example instances, and snmp mapping is provided in a single file: joint_all.sf

[0081] Some of the advantages provided by embodiments may be summarized as follows:

[0082] A one-to-one correspondence between a DN prototype and a MIB definition can be established. This correspondence can be independent of any particular instantiation of an information model. That is, Instance DN's (except for singletons) are not given pre-defined mappings: they are distinguished via their right-hand-side values. The MIB definition is comprised of both the definition of attributes as well as the object ID prefix (prior to any indexing).

[0083] It is possible (at runtime) to make a translation between a given DN and a MIB table row and vice-versa. From either the MIB table row or the DN, a given attribute can be indicated. Such an indication of an attribute is static (does not depend on index values/RDN values) and local to the object/row indicated by the DN/Object ID.

[0084] A default mapping of variable length strings in RDN "right-hand-side's" (rhs) to an index value is provided.

[0085] Embodiments also provide for a mapping using integer values in RDN rhs's. The use of this mapping instead of a default is indicated by the definition of the information model/MIB.

[0086] An integer typed attribute within an object may be used as an index value instead of using the RDN rhs. Similarly, multiple integer typed variables within an object may be used as multiple indices instead of the RDN rhs.

[0087] For SNMP "get" and "trap" commands, it is possible to map information model attribute values of type "SANAMET" (SAF Name Type) to one of: a complete object ID; or the portion of the object ID corresponding to the indices.

[0088] The Information Model to SNMP mapping may be stored outside of the Information Model Management System (IMMS), so as to reduce hardware requirements and/or avoid "clutter".

[0089] Embodiments enable the definition of the mapping from the prototype of the DN of notification object plus the notification type to a trap MIB. Furthermore, it is possible to map the DN's of notification object and notifying object to either complete OID's or the index portion of the OID's.

[0090] Embodiments also enable an IMMS to SNMP mapping for only a subset of the IMMS model.

[0091] Embodiments not only allow hierarchy in an information model description but also inherently map associations of related LDAP or SAF Name Type using the only available complex data structure in SNMP, MIB Tables.

[0092] Instead of having little MIB's in each of the managed network elements, embodiments have a dedicated, centralized server that can take some of the load off the managed elements and provide a single point of contact for the NMS (which is a client of the SNMP gateway). This may provide performance advantages for the total network management solution.

[0093] While specific embodiments have been described herein for purposes of illustration, various other modifications will be apparent to a person skilled in the art and may be made without departing from the scope of the invention.

Claims

1. A method of using the Simple Network Management Protocol, SNMP, to manage a Lightweight Directory Access Protocol, LDAP, information base comprising a plurality of instance entries each having one or more information model object attributes and each being identified by a respective Distinguished Name, DN, composed of type/value pairs; the method comprising mapping an SNMP Management Information Base, MIB, object that is identified by an Object Identifier, OID, composed of a series of sub-identifiers, to a DN and attribute of an instance entry of the LDAP information base; **characterized in that** this mapping comprises:

   (a) mapping a first sequence of sub-identifiers of the MIB object OID to a sequence of DN type/value pair types,
   (b) mapping a further sub-identifier of the MIB object OID to an attribute name, this sub-identifier corresponding to a scalar/columnar,
   (c) mapping a second sequence of sub-identifiers of the MIB object OID to a sequence of DN type/value pair values.

2. A method according to claim 1, wherein the mapping performed in (a) is carried out using a prototype DN in which each type/value pair maps a DN type to an OID sub identifier, the types in a prototype DN defining a set of DN instances for a given information model instance.

3. A method according to claim 2, wherein the prototype DN is associated with an instance object that provides a sub-identifier to attribute name map, this instance object being used to carry out the mapping in (b).

4. A method according to claim 1, wherein the mapping performed in (c) is carried out using a mapping function to convert the sub identifiers of the second sequence of sub-identifiers into corresponding DN type/value pair values.

5. An SNMP network management system (16) adapted to manage a LDAP information base comprising a plurality of instance entries each having one or

more object attributes and each being identified by a respective Distinguished Name composed of type/value pairs; the network management system being further adapted to map an SNMP MIB object that is identified by an Object Identifier, OID, composed of a series of sub-identifiers, to a DN and attribute of an instance entry of the LDAP information base; **characterized in that** the network management system is arranged to carry out this mapping by:

(a) mapping a first sequence of sub-identifiers of the MIB object OID to a sequence of DN type/value pair types,
(b) mapping a further sub-identifier of the MIB object OID to an attribute name, this sub-identifier corresponding to a scalar/columnar,
(c) mapping a second sequence of sub-identifiers of the MIB object OID to a sequence of DN type/value pair values.

6. A network management system (16) according to claim 5, wherein the mapping performed in (a) is arranged to be carried out using a prototype DN in which each type/value pair maps a DN type to an OID sub identifier, the types in a prototype DN defining a set of DN instances for a given information model instance.

7. A network management system (16) according to claim 6, wherein the prototype DN is associated with an instance object that provides a sub-identifier to attribute name map,the system being arranged to use this instance object to carry out the mapping in (b).

8. A network management system (16) according to claim 5, wherein system is arranged to carry out the mapping performed in (c) using a mapping function to convert the sub identifiers of the second sequence of sub-identifiers into corresponding DN type/value pair values.

9. A computer program comprising computer program means adapted to perform all of the steps of claim 1 when said program is run on a computer.

10. A computer program as claimed in claim 9 embodied on a computer readable medium.

Patentansprüche

1. Verfahren zur Verwendung des Simple Network Management Protocol, SNMP, zum Verwalten einer Informationsbasis gemäß dem Lightweight Directory Access Protocol, LDAP, aufweisend eine Vielzahl von Instanzeinträgen mit jeweils einem oder mehreren Informationsmodell-Objektattribut(en), das/die jeweils durch einen entsprechenden aus Typ/Wert-Paaren zusammengesetzten Distinguished Name, DN, identifiziert sind; wobei das Verfahren das Mapping eines Objekts der SNMP Management Information Base, MIB, das durch einen aus einer Reihe von Sub-Identifiern zusammengesetzten Object Identifier, OID, identifiziert ist, auf einen DN und ein Attribut eines Instanzeintrags der LDAP-Informationsbasis aufweist, **dadurch gekennzeichnet, dass** dieses Mapping Folgendes aufweist:

(a) Mapping einer ersten Sequenz von Sub-Identifiern des MIB-Objekt-OID auf eine Sequenz von DN-Typ/Wert-Paartypen,
(b) Mapping eines weiteren Sub-Identifiers des MIB-Objekt-OID auf einen Attributnamen, wobei dieser Sub-Identifier einem skalaren/spaltenweisen Identifier entspricht,
(c) Mapping einer zweiten Sequenz von Sub-Identifiern des MIB-Objekt-OID auf eine Sequenz von DN-Typ/Wert-Paartypen.

2. Verfahren nach Anspruch 1, wobei das in (a) durchgeführte Mapping unter Verwendung eines Prototyp-DN ausgeführt wird, wobei jedes Typ/Wert-Paar einen DN-Typ auf einen OID-Sub-Identifier abbildet und die Typen in einem Prototyp-DN eine Gruppe von DN-Instanzen für eine gegebene Informationsmodell-Instanz definieren.

3. Verfahren nach Anspruch 2, wobei der Prototyp-DN mit einem Instanzobjekt assoziiert ist, das einen Sub-Identifier für die Attributnamen-Map bereitstellt, wobei dieses Instanzobjekt zur Ausführung des Mapping in (b) verwendet wird.

4. Verfahren nach Anspruch 1, wobei das in (c) durchgeführte Mapping unter Verwendung einer Mapping-Funktion ausgeführt wird, um die Sub-Identifier der zweiten Sequenz von Sub-Identifiern in entsprechende DN-Typ/Wert-Paarwerte umzusetzen.

5. SNMP-Netzverwaltungsystem (16), eingerichtet zur Verwaltung einer LDAP-Informationsbasis, aufweisend eine Vielfalt von Instanzeinträgen, die jeweils ein oder mehrere Objektattribut(e) aufweisen und die jeweils identifiziert sind durch einen entsprechenden aus Typ/Wert-Paaren zusammengesetzten Distinguished Name; wobei das Netzverwaltungssystem weiter eingerichtet ist, um ein SNMP-MIB-Objekt, das durch einen aus einer Reihe von Sub-Identifiern zusammengesetzten Object Identifier, OID, identifiziert ist, auf einen DN und ein Attribut eines Instanzeintrags der LDAP-Informationsbasis abzubilden, **dadurch gekennzeichnet, dass** das Netzverwaltungssystem eingerichtet ist, um dieses Mapping wie folgt auszuführen:

(a) Mapping einer ersten Sequenz von Sub-Identifiern des MIB-Objekt-OID auf eine Sequenz von DN-Typ/Wert-Paartypen,
(b) Mapping eines weiteren Sub-Identifiers des MIB-Objekt-OID auf einen Attributnamen, wobei dieser Sub-Identifier einem skalaren/spaltenweisen Identifier entspricht,
(c) Mapping einer zweiten Sequenz von Sub-Identifiern des MIB-Objekt-OID auf eine Sequenz von DN-Typ/Wert-Paartypen.

6. Netzverwaltungssystem (16) nach Anspruch 5, wobei das in (a) durchgeführte Mapping für die Ausführung unter Verwendung eines Prototyp-DN eingerichtet ist, wobei jedes Typ/Wert-Paar einen DN-Typ auf einen OID-Sub-Identifier abbildet und die Typen in einem Prototyp-DN eine Gruppe von DN-Instanzen für eine gegebene Informationsmodell-Instanz definieren.

7. Netzverwaltungssystem (16) nach Anspruch 6, wobei der Prototyp-DN mit einem Instanzobjekt assoziiert ist, das einen Sub-Identifier für die Attributnamen-Map bereitstellt, wobei das System eingerichtet ist, um dieses Instanzobjekt zur Ausführung des Mappings in (b) einzusetzen.

8. Netzverwaltungssystem (16) nach Anspruch 5, wobei das System eingerichtet ist, um das in (c) durchgeführte Mapping unter Verwendung einer Mapping-Funktion auszuführen, um die Sub-Identifier der zweiten Sequenz von Sub-Identifiern in entsprechende DN-Typ/Wert-Paarwerte umzusetzen.

9. Computerprogramm, das Computerprogrammmittel aufweist, die eingerichtet sind zur Durchführung aller Schritte von Anspruch 1, wenn das Programm auf einem Computer ausgeführt wird.

10. Computerprogramm nach Anspruch 9, ausgeführt auf einem computerlesbaren Medium.

Revendications

1. Procédé d'utilisation du protocole simple de gestion de réseau, SNMP, pour gérer un protocole d'accès au répertoire allégé, LDAP, une base de données comprenant une pluralité d'entrées d'instance ayant chacune un ou plusieurs attributs d'objets de modèles d'informations et étant chacune identifiée par un nom distinctif respectif, DN, composé de paires type/valeur ; le procédé comprenant la mise en correspondance d'une base d'informations de gestion SNMP, MIB, objet qui est identifié par un identifiant d'objet, OID, constitué d'une série de sous-identifiants, avec un DN ou attribut d'une entrée d'instance de la base d'informations LDAP ; **caractérisé en ce que** cette mise en correspondance comprend :

(a) la mise en correspondance d'une première séquence de sous-identificateurs de l'objet MIB, OID, avec une séquence de types de paires type/valeur DN,
(b) la mise en correspondance d'un autre sous-identifiant de l'objet MIB, OID, avec un nom d'attribut, ce sous-identifiant correspondant à un scalaire ou une colonne,
(c) la mise en correspondance d'une seconde séquence de sous-identifiants de l'objet MIB, OID, avec une séquence de valeurs de paires type/valeur DN.

2. Procédé selon la revendication 1, dans lequel la mise en correspondance effectuée en (a) est effectuée à l'aide d'un prototype DN dans lequel chaque paire type/valeur met en correspondance un type DN avec un sous-identifiant OID, les types d'un prototype DN définissant un ensemble d'instances DN pour une instance donnée de modèle d'informations.

3. Procédé selon la revendication 2, dans lequel le prototype DN est associé à un objet d'instance qui produit un sous-identifiant à la correspondance nom d'attribut, cet objet d'instance étant utilisé pour effectuer la mise en correspondance dans (b).

4. Procédé selon la revendication 1, dans lequel la mise en correspondance effectuée en (c) est effectuée à l'aide d'une fonction de mise en correspondance pour convertir les sous-identifiants de la seconde séquence de sous-identifiants en valeurs de paires type/valeur DN correspondantes.

5. Système de gestion de réseau SNMP (16) adapté pour gérer une base d'informations LDAP comprenant une pluralité d'entrées d'instance ayant un ou plusieurs attributs d'objet et étant chacune identifiée par un nom distinctif respectif composé de paires type/valeur ; le système de gestion de réseau étant en outre adapté pour mettre en correspondance un objet MIB SNMP qui est identifié par un identifiant d'objet, OID, constitué d'une série de sous-identifiants, avec un DN et attribut d'une entrée d'instance de la base d'informations LDAP ; **caractérisé en ce que** le système de gestion de réseau est adapté pour réaliser cette mise en correspondance par :

(a) la mise en correspondance d'une première séquence de sous-identificateurs de l'objet MIB, OID, avec une séquence de types de paires type/valeur DN,
(b) la mise en correspondance d'un autre sous-identifiant de l'objet MIB OID avec un nom d'attribut, ce sous-identifiant correspondant à un scalaire ou une colonne,

(c) la mise en correspondance d'une seconde séquence de sous-identifiants de l'objet MIB OID avec une séquence de valeurs de paires type/valeur DN.

6. Système de gestion de réseau (16) selon la revendication 5, dans lequel la mise en correspondance effectuée en (a) est adaptée pour être effectuée à l'aide d'un prototype DN dans lequel chaque paire type/valeur met en correspondance un type DN avec un sous-identifiant OID, les types d'un prototype DN définissant un ensemble d'instances DN pour une instance données de modèle d'informations.

7. Système de gestion de réseau (16) selon la revendication 6, dans lequel le prototype DN est associé à un objet d'instance qui produit un sous-identifiant à la correspondance nom d'attribut, le système étant adapté pour utiliser cet objet d'instance pour effectuer la mise en correspondance en (b).

8. Système de gestion de réseau (16) selon la revendication 5, dans lequel le système est adapté pour effectuer la mise en correspondance effectuée en (c) à l'aide d'une fonction de mise en correspondance pour convertir les sous-identifiants de la seconde séquence de sous-identifiants en valeurs de paires type/valeur DN correspondantes.

9. Programme d'ordinateur comprenant des moyens de programme d'ordinateur adaptés pour réaliser toutes les étapes de la revendication 1 lorsque ledit programme est exécuté sur un ordinateur.

10. Programme d'ordinateur selon la revendication 9 intégré sur un support lisible par ordinateur.

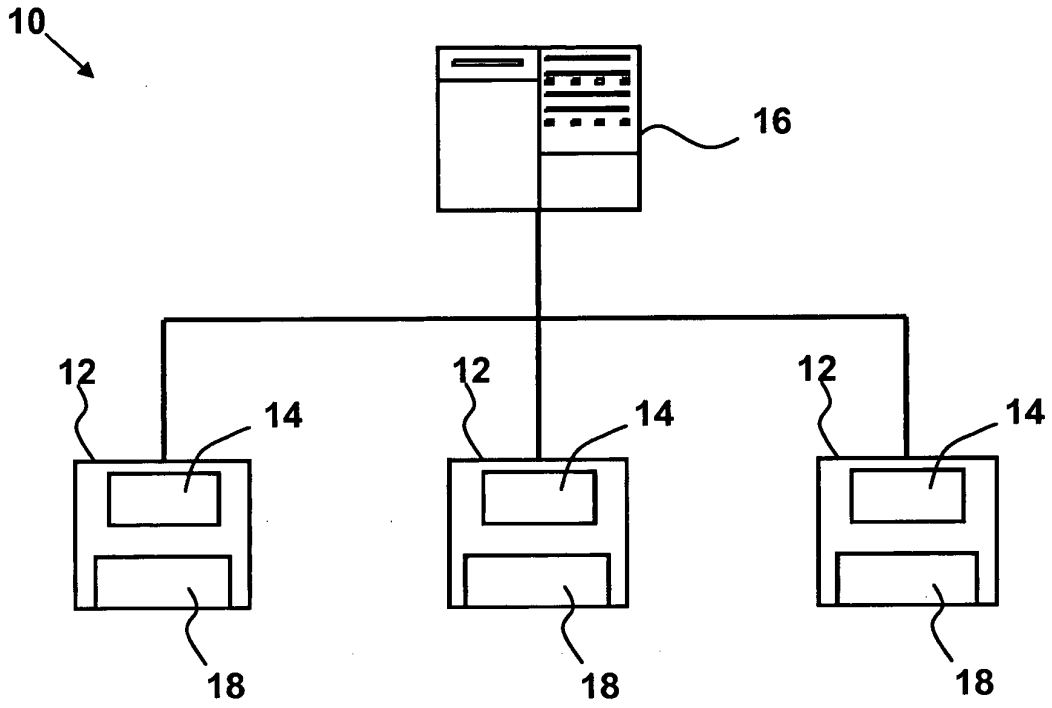

Figure 1

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 20030208609 A1 **[0004]**
- US 20050080886 A1 **[0005]**

Non-patent literature cited in the description

- **MAZUMDAR S.** DIRECTORY ENABLED MANAGEMENT INFORMATION BASE - INTEGRATION OF MIB WITH DIRECTORIES USING COSNAMING AND JNDI. *IFIP/IEEE INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS:OPERATIONS AND MANAGEMENT,* 25 August 2000, vol. XX, XX, 1-32 **[0006]**